ARPA Network Information Center
Stanford Research Institute
Menlo Park, California 94025

NIC #13877
Packet Radio Temporary Note #18

Richard Binder
University of Hawaii
January 23, 1973

GENERAL DESCRIPTION OF THE UNIVERSITY OF HAWAII ALOHA SYSTEM MENEHUNE

INTRODUCTION

The MENEHUNE (for legendary Hawaiian elf) is the minicomputer used to interface the ALOHA channel to an IBM 360/65 time-sharing system (TSO). ALOHA SYSTEM internal documents CCG/G-1, CCG/G-2, CCG/G-3, and reports B72-2, B72-3 should be consulted for descriptions of related interfaces and protocol.

MENEHUNE FUNCTIONS

Figure 1 shows the MENEHUNE (an HP 2115A minicomputer) in relation to the rest of the system. The IBM 1827 is a front-end unit allowing connection of analog and digital devices to the 360. While its use is not essential, its existence provided a simple means of physically connecting the MENEHUNE to the 360.

The MENEHUNE communicates full-duplex with the ensemble of radio users, receiving from users on frequency f_1 while simultaneously transmitting to the users on another frequency f_2. The random-accessing technique is used on receiving channel f_1, while messages are queued by the MENEHUNE for consecutive transmission to the users on outgoing channel f_2. Messages are sent in either direction in the form of packets. Each packet includes a header containing the terminal address (ID) and other information, and two parity check words for error control (Figure 2).

As a packet is received serially at the Encoder-Decoder, each 16-bit increment is transfered in parallel to the MENEHUNE, which has a 16-bit word size. The MENEHUNE must accept and store each of these words as they arrive, and perform checks at certain times as described below. If no errors are detected after the entire packet has been received, the MENEHUNE then sends an acknowledgement message to the terminal and forwards the message to the 360. If errors are detected, the MENEHUNE discards the message and no

Figure 1. SYSTEM ORGANIZATION

ISR Devices
DIU
Network
Decoder
Encoder
Modem
Receiver
Transmitter

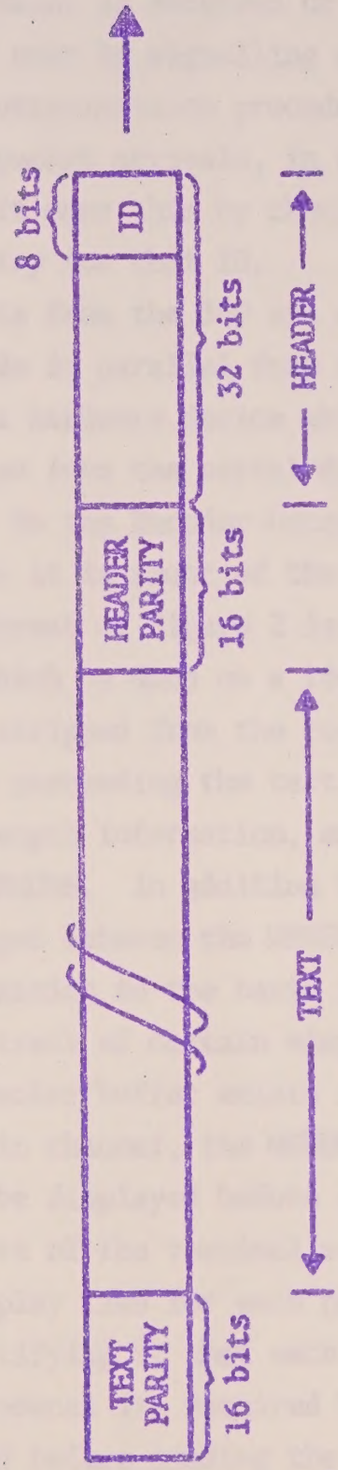
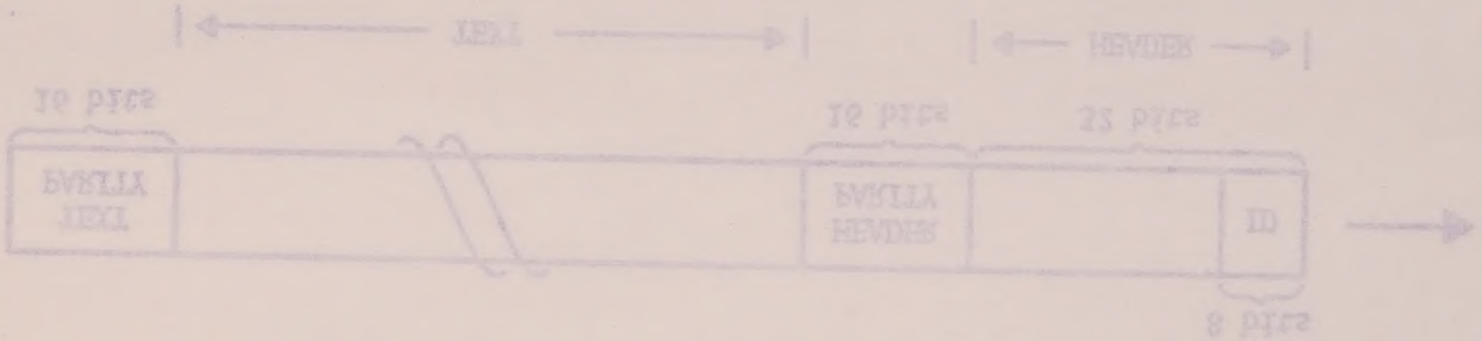


Figure 2. RADIO PACKET FORMAT


TEXT
HEADER
16 bits
16 bits
25 bits
PARITY
TEXT
PARITY
HEADER
ID
8 bits

acknowledgement is sent. The absence of an acknowledgement after a specified time causes a terminal to automatically retransmit the packet, which it has held in a buffer. The terminal will repeat this process until either an acknowledgement is received or it has tried three times, after which it will notify the user by signalling an alarm.

The retransmission procedure requires that the MENEHUNE also check for duplicate packet arrivals, in order that it not send the same message twice to the 360. It does this by checking a special bit in the packet header against a table entry for that ID.

Packets from the 360 are queued and sent to terminals by transferring 16-bit words in parallel from the MENEHUNE to the Encoder-Decoder. The latter is a hardware device which generates the required parity words and inserts them into the serial data stream passed to the modem. Prior to passing the packet to the Encoder-Decoder, the MENEHUNE generates the required header and inserts it in front of the text.

The format of Figure 2 is not used in transferring packets to and from the 360, which is done on a 16-bit parallel basis. The header and parity words are stripped from the received packet and a single 16-bit word is sent to the 360 preceeding the text. This word contains the ID of the terminal and text length information, and also preceeds the packets sent from the 360 to the MENEHUNE. In addition to the message packets, 16-bit status messages are exchanged between the MENEHUNE and 360 for bookkeeping purposes.

In addition to the basic functions just described, the MENEHUNE must also keep track of certain elapsed times for each terminal. First, since only a single packet buffer exists in each terminal's hardware interfacing it to the radio channel, the MENEHUNE must allow time for all characters of a packet to be displayed before sending the next packet. Thus it must know the display rate of the terminal associated with each ID and count off the required display time for each packet sent. Since the 360 demands a status message notifying it that each message transaction has been completed, the MENEHUNE reduces its required buffer space by waiting until the display time has elapsed before sending the 360 the completion status message.

A second timing requirement arises from the radio equipment used at the terminals. Some of the terminals use half-duplex equipment, which requires a minimum waiting time after transmitting before beginning to receive. For certain of the equipment used this time is on the order of 100 milliseconds,

Digitized by the Internet Archive
in 2025 with funding from
Amateur Radio Digital Communications, Grant 151

https://archive.org/details/temporary-note-018

whereas the typical time taken by the MENEHUNE to respond with an acknowledgement message for a received packet is less than one millisecond. Thus the MENEHUNE must count off the 100 milliseconds before sending the acknowledgement, while remaining free to send and receive to other terminals.

Some additional nonessential functions have been implemented to make life easier for the users. For example:

- the MENEHUNE acts as a watchdog timer on the 360, allowing it to detect when the 360 is down; when it is the MENEHUNE sends a "down" message in response to packets from users, allowing them to immediately know the status of the system;
- the MENEHUNE allows any two terminals to link together in order to receive each other's input and/or output while one or both are connected to the 360;
- a user can change the display rate used by the MENEHUNE to count off his terminal display time by sending a special message to the MENEHUNE.

PROGRAM STRUCTURE

Organization

The HP 2115A minicomputer has a core storage of 8K 16-bit words and a cycle time of 2 microseconds. It provides eight I/O ports and two direct-memory-access (DMA) channels, with a hardware priority interrupt system. Core is divided into 1024-word pages; the first page is directly addressable from all other pages, and indirect addressing may be used when required for referencing words on other pages. A special compiler was written at the University of Hawaii to allow programs to be written in the XPL language, a subset of PL/1. The compiler runs on the IBM 360/65 and produces the HP binary machine code as object code, which is stored on a disk for subsequent transfer to the MENEHUNE.

The program is interrupt-driven, and consists of three groups of subroutines: (1) interrupt subroutines, (2) flagged subroutines, and (3) called subroutines (Figure 3). The interrupt subroutines are entered as the result

whereas the typical time taken by the MENEHUNE to respond with an acknowledgement message for a received packet is less than one millisecond. Thus the MENEHUNE must count off the 100 milliseconds before sending the acknowledgement, while remaining free to send and receive to other terminals.

Some additional nonessential functions have been implemented to make life easier for the users. For example:

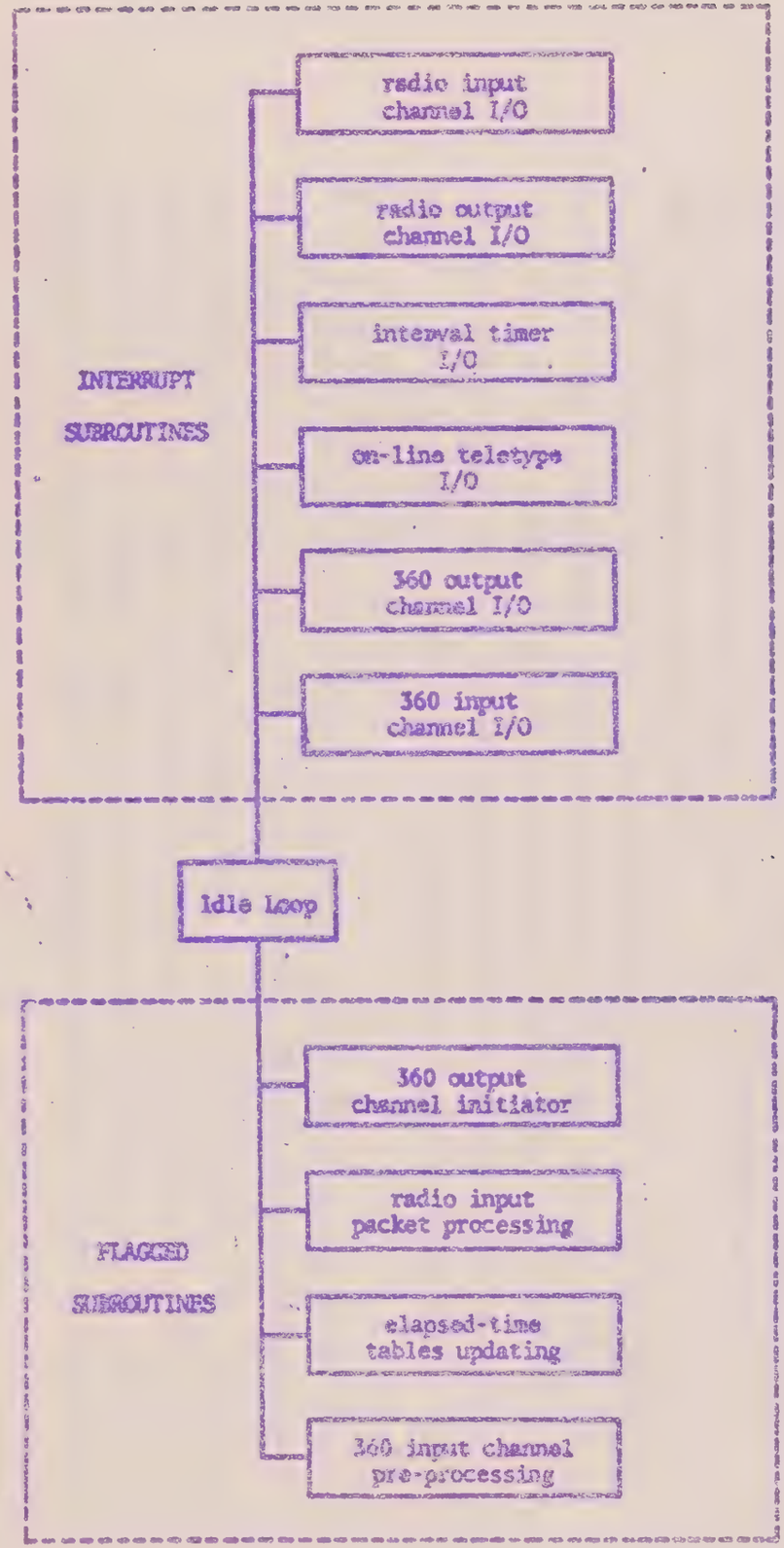
- the MENEHUNE acts as a watchdog timer on the 360, allowing it to detect when the 360 is down; when it is the MENEHUNE sends a "down" message in response to packets from users, allowing them to immediately know the status of the system;
- the MENEHUNE allows any two terminals to link together in order to receive each other's input and/or output while one or both are connected to the 360;
- a user can change the display rate used by the MENEHUNE to count off his terminal display time by sending a special message to the MENEHUNE.

PROGRAM STRUCTURE

Organization

The HP 2115A minicomputer has a core storage of 8K 16-bit words and a cycle time of 2 microseconds. It provides eight I/O ports and two direct-memory-access (DMA) channels, with a hardware priority interrupt system. Core is divided into 1024-word pages; the first page is directly addressable from all other pages, and indirect addressing may be used when required for referencing words on other pages. A special compiler was written at the University of Hawaii to allow programs to be written in the JOVIAL language, a subset of PL/1. The compiler runs on the IBM 360/65 and produces the HP binary machine code as object code, which is stored on a disk for subsequent transfer to the MENEHUNE.

The program is interrupt-driven, and consists of three groups of subroutines: (1) interrupt subroutines, (2) flagged subroutines, and (3) called subroutines (Figure 3). The interrupt subroutines are entered as the result



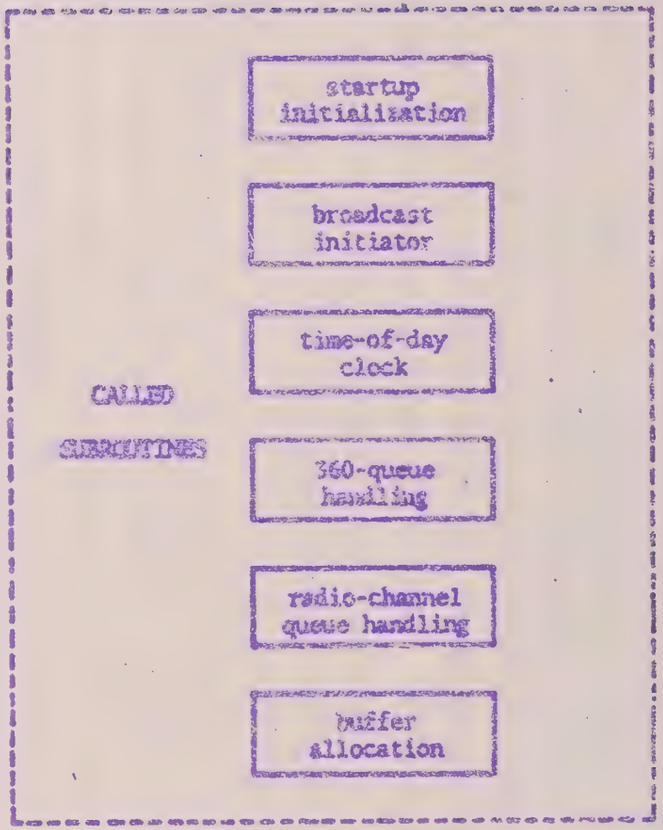


Figure 3. PROGRAM STRUCTURE

of an external interrupt, and when exited control is returned to the program instruction which was about to be executed prior to the interrupt. When an event requiring further processing has taken place, such as the last word received of a packet, a flag is set by an instruction in the appropriate interrupt subroutine which causes one of the flagged subroutines to be awakened. This awakening takes place as follows: when no subroutines are active, the program executes an idle loop which repeatedly checks a special flag word; when a bit of the flag becomes logical one, the idle loop then calls the flagged subroutine associated with that bit; when the flagged subroutine has finished its processing, control is returned to the idle loop. The "called subroutines" may be called directly by any of the interrupt, flagged, or other called subroutines. They provide such functions as inserting and removing packets to and from queues, performing dynamic buffer allocation, maintaining a time-of-day clock, and initializing tables and variables whenever the program is restarted.

Timing Considerations

To accept an incoming message from the radio channel, the MENEHUNE must both identify the start of each new message and be ready to accept each new 16-bit word within the time determined by the channel transmission rate (660 microseconds for a rate of 24Kb/s). This same time constraint exists for words of outgoing messages also, except that the MENEHUNE can start each outgoing message at a chosen time.

The exchange with the 360, on the other hand, involves a hand-shaking procedure which allows each computer to proceed with the next word when it is ready, with the other computer guaranteed to wait. The other two I/O connections consist of an interval timer card which generates an interrupt every 1.14 milliseconds, and an on-line teletype used for program debugging (and as a time-sharing terminal).

Figure 3 reflects these time constraints in the vertical ordering of the subroutines. The radio input channel has highest priority, followed by the radio output channel, the interval timer, the on-line teletype, and the 360 I/O. The next level of the program consists of the idle loop, followed by the flagged subroutines. Any interrupt subroutine may interrupt any subroutine below it in the figure. A flagged subroutine is entered before any

below it in the figure if both happen to be flagged at the same time. However, once a flagged subroutine has been entered it runs to completion before another flagged subroutine (above or below it) can be entered.

Display and acknowledgement times are handled by special tables whose processing is initiated by the interval timer. Whenever the interrupt subroutine associated with the timer counts to ten milliseconds, it awakens a flagged subroutine which maintains the tables. Entries in these tables are set to zero for all inactive ID's; those in the process of counting off time have an entry greater than zero. Each time the flagged subroutine is called it decrements all non-zero entries; when a decremented entry becomes zero an action is taken appropriate to the table. Entries are activated by the interrupt or other flagged subroutines as a result of some event. For example, the radio output channel interrupt subroutine enters a value in the display time table after the last 16-bit word of a packet has been sent; this value is then decremented to zero by the subsequent flagged calls via the interval time.

The parallel-processing environment created by the interrupt structure of the program requires care in the use of common subroutines and tables. To overcome these problems the present program either turns off the interrupt system for the duration of the shared access or, in some cases, uses duplicate code.

Buffer Requirements

Except for one complicating factor to be discussed shortly, the buffer requirements are small. The shortest length packet is about 370 bits, including header and parity information. At 24Kb/s a minimum of 15 milliseconds is required for each incoming packet, while it takes on the order of one millisecond to transfer a packet to the 360. Thus, slightly more than one maximum packet length buffer (80 characters) dedicated to the incoming channel is sufficient for worst case (zero inter-packet time) arrivals.

Buffer requirements for outgoing packets are determined by two factors: (1) the rate of arrivals from the 360 relative to the rate of transmission over the radio channel, and (2) the method of handling errors in packets arriving at user terminals. The rate of 360 arrivals is controlled by the MENEHUNE via the handshaking procedure; if it does not have a buffer available for a new 360 packet, it simply does not begin accepting the packet until one becomes

available. Thus only one maximum-size buffer is required for the outgoing channel, ignoring error control procedures. The latter, however, can have a profound effect on the buffer requirements. In our present system, no retransmissions are made to terminals whether the packet is received with errors or not; thus the single buffer fulfills the requirements. If a positive acknowledgement were sent from each terminal for each error-free packet, the MENEHUNE would have to hold each packet for possible retransmission until the acknowledgement is received, and in general could be waiting for acknowledgements from several different terminals at one time. A negative acknowledgement scheme with a packet sequence number modulus of say, 16, could require the MENEHUNE to hold the last 16 packets sent to each terminal for possible retransmission.

The complicating factor mentioned at the beginning of this subsection consists of the need to delay the acknowledgement message sent to half-duplex radio terminals. For a turnaround time of 100 ms and minimum packet time of 15 ms, 8 buffers are required to accept all arrivals in the worst case. Most of these could be eliminated by forwarding the packet to the 360 as soon as it is received, rather than after the acknowledgement has been sent. However, since the response to the terminal would most likely arrive from the 360 before the 100 ms was up, this would complicate the outgoing channel (requiring either extra buffering at that time or a more complex mechanism for passing packets from the 360). Fortunately, the problem is mitigated by two factors: (1) the newer solid-state transceivers used in the system have a turnaround time on the order of 1 ms; and (2) if the MENEHUNE does not accept an arriving packet (for any reason), the terminal will automatically retransmit it after a specified time. Thus, insufficient buffer space in the MENEHUNE can result in delays experienced by the user, but will not cause a loss of data.

Some of the system frills, such as terminal-to-terminal linking, can significantly increase the buffer requirements. The increase depends on the number of terminals which might be receiving linked output or other special messages at one time.

CONCLUSION

Current plans include connection of the MENEHUNE to the ARPANET. The

available. Thus only one maximum-size buffer is required for the outgoing channel, ignoring error control procedures. The latter, however, can have a profound effect on the buffer requirements. In our present system, no retransmissions are made to terminals whether the packet is received with errors or not; thus the single buffer fulfills the requirements. If a positive acknowledgement were sent from each terminal for each error-free packet, the MEMEBUNE would have to hold each packet for possible retransmission until the acknowledgement is received, and in general could be waiting for acknowledgements from several different terminals at one time. A negative acknowledgement scheme with a packet sequence number modulus of say, 16, could require the MEMEBUNE to hold the last 16 packets sent to each terminal for possible retransmission.

The complicating factor mentioned at the beginning of this subsection consists of the need to delay the acknowledgement message sent to half-duplex radio terminals. For a turnaround time of 100 ms and minimum packet time of 15 ms, 8 buffers are required to accept all arrivals in the worst case. Most of these could be eliminated by forwarding the packet to the 360 as soon as it is received, rather than after the acknowledgement has been sent. However, since the response to the terminal would most likely arrive from the 360 before the 100 ms was up, this would complicate the outgoing channel (requiring either extra buffering at that time or a more complex mechanism for passing packets from the 360). Fortunately, the problem is mitigated by two factors: (1) the newer solid-state transceivers used in the system have a turnaround time on the order of 1 ms; and (2) if the MEMEBUNE does not accept an arriving packet (for any reason), the terminal will automatically retransmit it after a specified time. Thus, insufficient buffer space in the MEMEBUNE can result in delays experienced by the user, but will not cause a loss of data.

Some of the system frills, such as terminal-to-terminal linking, can significantly increase the buffer requirements. The increase depends on the number of terminals which might be receiving linked output or other special messages at one time.

CONCLUSION

Current plans include connection of the MEMEBUNE to the ARPANET. The

MENEHUNE will act in a store-and-forward capacity, routing incoming radio packets to the local ARPANET interface for subsequent satellite routing to its mainland destination, and conversely. Radio users will notify the MENEHUNE they wish to connect to the ARPANET by use of a special command, and subsequent packets will be routed as just described until the user breaks the connection with another special command. Other satellite networks being planned for the Pacific area will probably be interfaced to the MENEHUNE in a similar manner.

We are also replacing the HP 2115A with an HP 2100, primarily to obtain more core for the expanded program functions. The use of the 360 to generate and load new versions of the MENEHUNE's program make these and other changes a relatively easy task.

MENEHUNE will act in a store-and-forward capacity, routing incoming radio packets to the local ARPANET interface for subsequent satellite routing to its mainland destination, and conversely. Radio users will notify the MENEHUNE they wish to connect to the ARPANET by use of a special command, and subsequent packets will be routed as just described until the user breaks the connection with another special command. Other satellite networks being planned for the Pacific area will probably be interfaced to the MENEHUNE in a similar manner.

We are also replacing the HP 2115A with an HP 2100, primarily to obtain more core for the expanded program functions. The use of the 560 to generate and load new versions of the MENEHUNE's program make these and other changes a relatively easy task.